- A brief overview
- Key tools and concepts used
- A link to the source code or a complete snippet you can run and modify

By the end of this tutorial, you'll have a collection of practical JavaScript mini-projects that you can proudly showcase in your portfolio or GitHub profile.

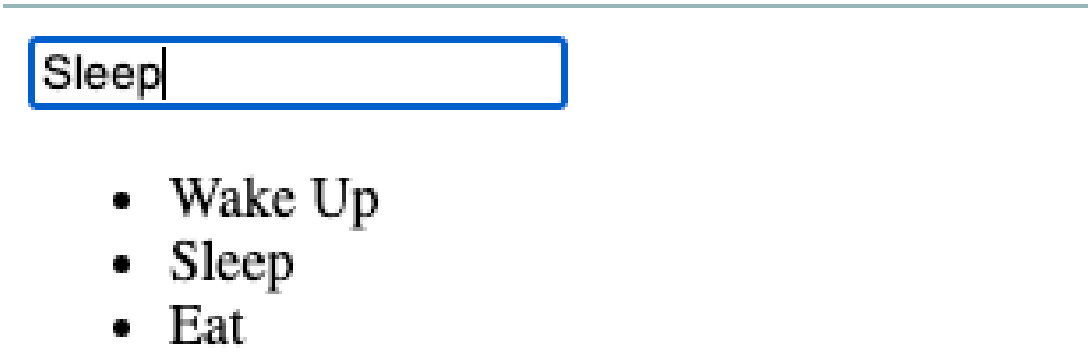
Let's get started and turn your JavaScript knowledge into real, working applications!

1. To-Do List App

Description: A task management app to add, complete, and remove tasks, stored locally in the browser.

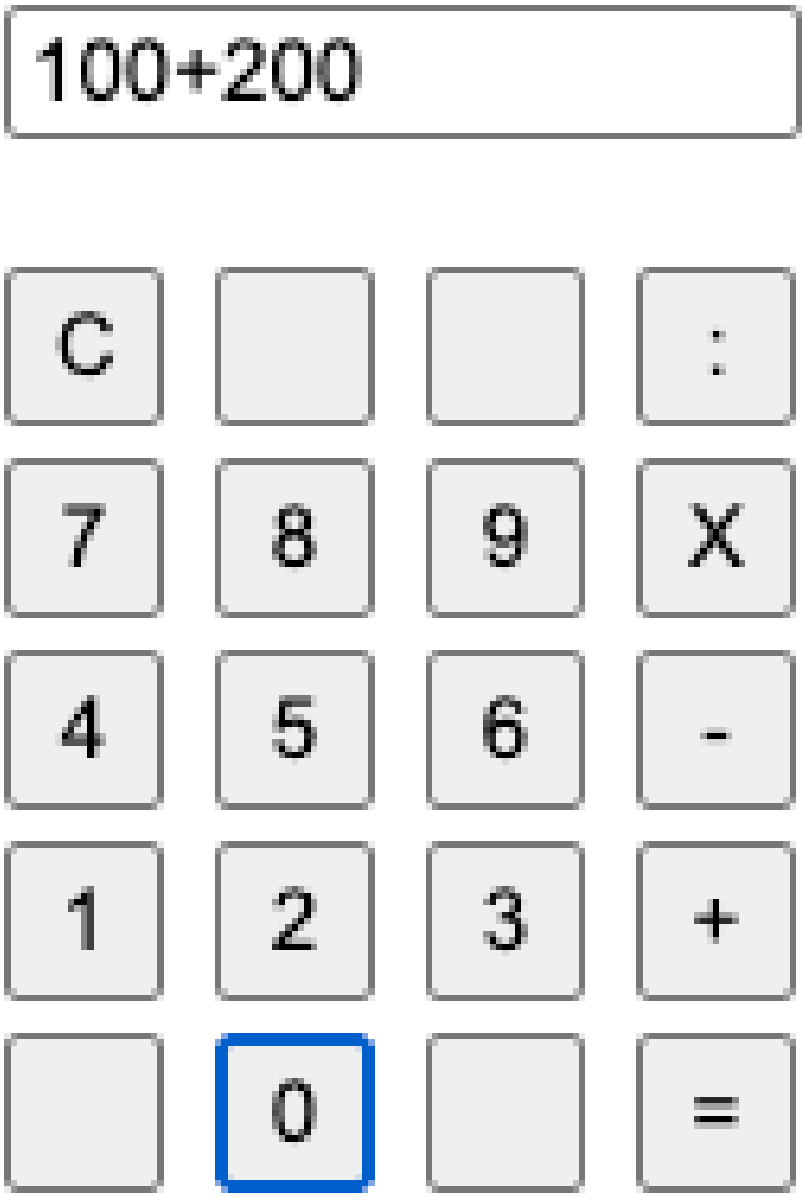
Tools & Concepts: DOM manipulation, Event listeners, localStorage

```
<input id="todo-input" placeholder="Add a task" />
<ul id="todo-list"></ul>

<script>
  const input = document.getElementById("todo-input");
  const list = document.getElementById("todo-list");

  input.addEventListener("keypress", (e) => {
    if (e.key === "Enter" && input.value.trim()) {
      const li = document.createElement("li");
      li.textContent = input.value;
      li.onclick = () => li.remove();
      list.appendChild(li);
      input.value = "";
    }
  });
</script>
```



```
<input id="display" readonly />
<div style="margin: 10px">
  <div class="button-group">
    <button onclick="clearDisplay()">C</button>
    <button> </button>
    <button> </button>
    <button onclick="append(':')">:</button>
  </div>
  <div class="button-group">
    <button onclick="append('7')">7</button>
    <button onclick="append('8')">8</button>
    <button onclick="append('9')">9</button>
    <button onclick="append('X')">X</button>
  </div>
  <div class="button-group">
    <button onclick="append('4')">4</button>
    <button onclick="append('5')">5</button>
    <button onclick="append('6')">6</button>
    <button onclick="append('-')">-</button>
  </div>
  <div class="button-group">
    <button onclick="append('1')">1</button>
    <button onclick="append('2')">2</button>
    <button onclick="append('3')">3</button>
    <button onclick="append('+')">+</button>
  </div>
  <div class="button-group">
    <button> </button>
    <button onclick="append('0')">0</button>
    <button> </button>
    <button onclick="countValue()">=</button>
  </div>
</div>

<style>
  button {
    width: 25px;
    height: 25px;
    margin-right: 4px;
  }

  .button-group {
    display: block;
    padding-bottom: 5px;
  }

  input {
    margin: 10px;
    width: 125px;
  }
</style>

<script>
  const display = document.getElementById("display");

  function countValue() {
```

```
    if (display.value.includes("+")) display.value = eval(display.value);

    if (display.value.includes("-")) display.value = eval(display.value);

    if (display.value.includes(":"))
      display.value = display.value.split(":")[0] / display.value.split(":")[1];

    if (display.value.includes("X"))
      display.value = display.value.split("X")[0] * display.value.split("X")[1];
  }

  function append(value) {
    if (
      (value.includes("+") ||
        value.includes("-") ||
        value.includes(":") ||
        value.includes("X")) &&
      (display.value.includes("+") ||
        display.value.includes("-") ||
        display.value.includes(":") ||
        display.value.includes("X"))
    ) {
      alert("Limited to 2 values!");
      display.value = display.value.slice(0, -1);
    } else display.value += value;
  }

  function clearDisplay() {
    display.value = "";
  }
</script>
```



3. Digital Clock

Description: A live clock that updates every second using `setInterval`.

Tools & Concepts: `Date`, `setInterval`, string formatting

```
<div id="clock"></div>

<style>
  div {
    margin: 50px;
    font-size: 32px;
    font-family: "Courier New", Courier, monospace;
    font-weight: bold;
  }
</style>

<script>
  function updateClock() {
    const now = new Date();
    const time = now.toLocaleTimeString();
    document.getElementById("clock").textContent = time;
  }

  setInterval(updateClock, 1000);
  updateClock(); // initial call
</script>
```

9:15:51 AM

4. Weather App (Using API)

Description: Fetches the current weather based on the city name from the OpenWeatherMap API.

Tools & Concepts: Fetch API, working with JSON, input handling

```
<input id="city" placeholder="Enter city" />
<button onclick="getWeather()">Get Weather</button>
<div id="weather-result"></div>

<script>
  const API_KEY = "your_api_key_here";

  async function getWeather() {
    const city = document.getElementById("city").value;
    const res = await fetch(
      `https://api.openweathermap.org/data/2.5/weather?
q=${city}&units=metric&appid=${API_KEY}`
    );
    const data = await res.json();
    document.getElementById("weather-result").textContent =
      `${data.name}: ${data.main.temp}°C, ${data.weather[0].description}`;
  }
</script>
```

5. Quote Generator

Description: Generates a random inspirational quote from an API.

Tools & Concepts: Fetch API, async/await, innerText

```
<button onclick="getQuote()">New Quote</button>
<blockquote id="quote"></blockquote>

<script>
  async function getQuote() {
    const res = await fetch("https://api.quotable.io/random");
    const data = await res.json();
    document.getElementById("quote").textContent = `"${data.content}" —
${data.author}`;
  }

  getQuote(); // load initial quote
</script>
```

6. Number Guessing Game

Description: The browser picks a random number between 1 and 100. The player must guess it, with hints if the guess is too high or low.

Tools & Concepts: Random numbers, conditionals, input handling

```
<input id="guess" placeholder="Enter a number (1-100)" />
<button onclick="checkGuess()">Guess</button>
<p id="feedback"></p>

<script>
  const secret = Math.floor(Math.random() * 100) + 1;

  function checkGuess() {
    const guess = Number(document.getElementById("guess").value);
    const feedback = document.getElementById("feedback");
    if (guess === secret) {
      feedback.textContent = "🎉 Correct! You guessed it!";
    } else if (guess < secret) {
      feedback.textContent = "📉 Too low. Try again!";
    } else {
      feedback.textContent = "📈 Too high. Try again!";
    }
  }
</script>
```

7. Image Slider

Description: A basic image carousel that cycles through images with "Previous" and "Next" buttons.

Tools & Concepts: Arrays, DOM manipulation, event listeners

```
<img id="slider" width="300" />
<br/>
<button onclick="prev()">Previous</button>
<button onclick="next()">Next</button>

<script>
  const images = ["img1.jpg", "img2.jpg", "img3.jpg"];
  let index = 0;

  function showImage() {
    document.getElementById("slider").src = images[index];
  }

  function next() {
    index = (index + 1) % images.length;
    showImage();
  }

  function prev() {
    index = (index - 1 + images.length) % images.length;
    showImage();
  }

  showImage();
</script>
```

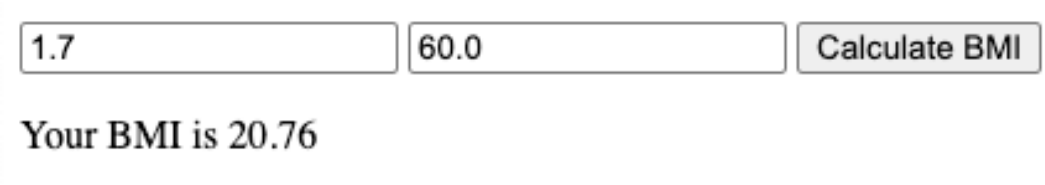
Previous Next

8. BMI Calculator

Description: Takes height and weight from the user and calculates their Body Mass Index.

Tools & Concepts: Form input, simple math, string formatting

```
<input id="height" placeholder="Height in meters" />
<input id="weight" placeholder="Weight in kg" />
<button onclick="calculateBMI()">Calculate BMI</button>
<p id="bmi-result"></p>

<script>
  function calculateBMI() {
    const h = parseFloat(document.getElementById("height").value);
    const w = parseFloat(document.getElementById("weight").value);
    const bmi = w / (h * h);
    document.getElementById("bmi-result").textContent =
      `Your BMI is ${bmi.toFixed(2)}`;
  }
</script>
```

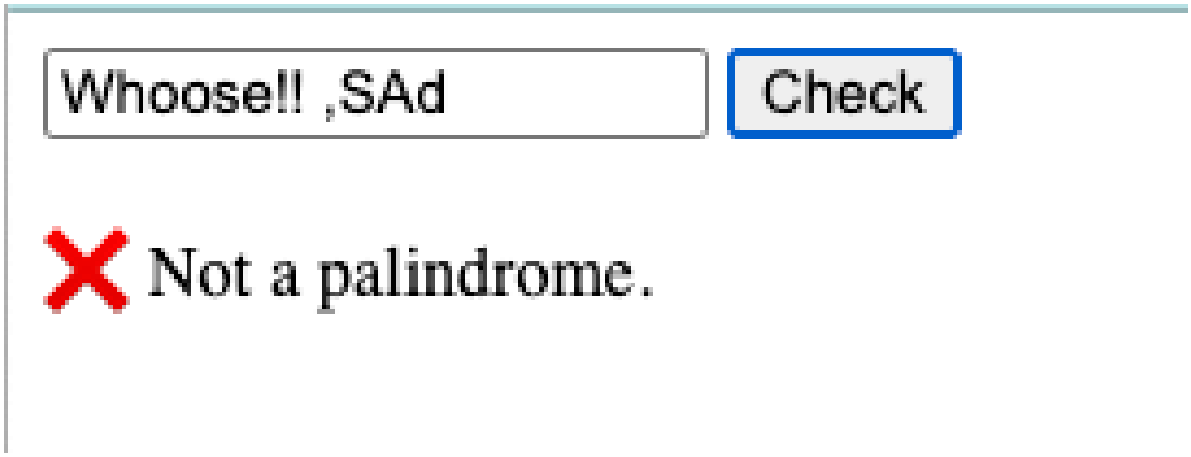


9. Palindrome Checker

Description: Checks whether the input word is a palindrome (reads the same backward).

Tools & Concepts: String manipulation, conditionals

```
<input id="word" placeholder="Enter a word" />
<button onclick="checkPalindrome()">Check</button>
<p id="pal-result"></p>

<script>
  function checkPalindrome() {
    const str = document.getElementById("word").value.toLowerCase();
    const reversed = str.split("").reverse().join("");
    const isPal = str === reversed;
    document.getElementById("pal-result").textContent =
      isPal ? "✔️ It's a palindrome!" : "❌ Not a palindrome.";
  }
</script>
```



10. Random Password Generator

Description: Generates a random password based on length and character sets.

Tools & Concepts: Random selection, string concatenation, loops

```
<input id="length" type="number" placeholder="Password length" />
<button onclick="generatePassword()">Generate</button>
<p id="password"></p>

<script>
  function generatePassword() {
    const len = document.getElementById("length").value;
    const chars =
"abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ0123456789!@#$%";
    let pass = "";
    for (let i = 0; i < len; i++) {
      pass += chars.charAt(Math.floor(Math.random() * chars.length));
    }
    document.getElementById("password").textContent = pass;
  }
</script>
```

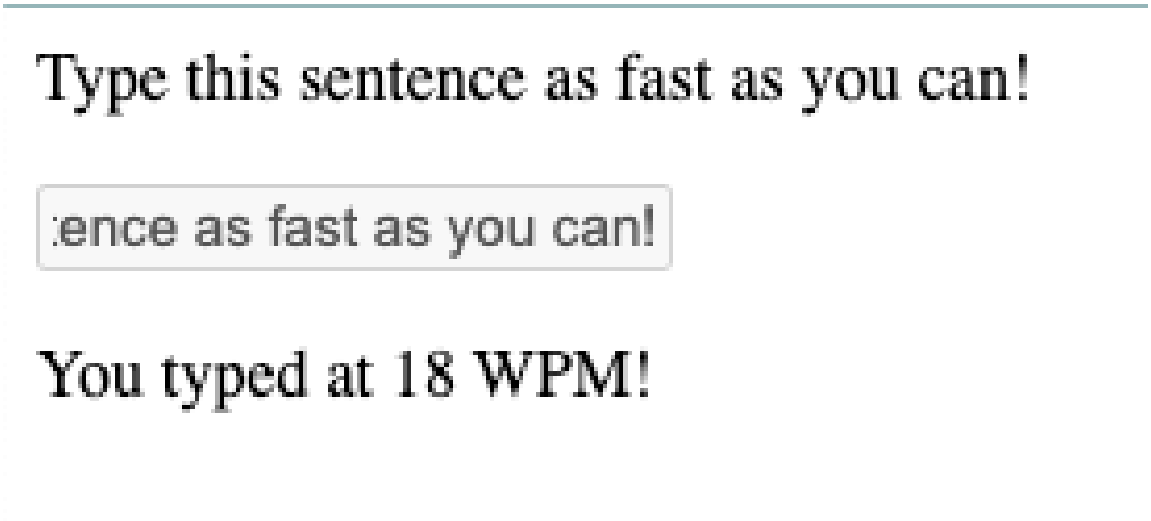
---

11. Typing Speed Test

Description: Measures how fast the user can type a displayed sentence, calculating words per minute (WPM).

Tools & Concepts: `Date` object, event handling, basic math

```
<p id="text">Type this sentence as fast as you can!</p>
<input id="inputText" placeholder="Start typing..." />
<p id="result"></p>

<script>
  const text = document.getElementById("text").textContent;
  const input = document.getElementById("inputText");
  let startTime;

  input.addEventListener("focus", () => {
    startTime = new Date().getTime();
  });

  input.addEventListener("input", () => {
    if (input.value === text) {
      const endTime = new Date().getTime();
      const timeTaken = (endTime - startTime) / 1000;
      const words = text.split(" ").length;
      const wpm = Math.round((words / timeTaken) * 60);
      document.getElementById("result").textContent = `You typed at ${wpm} WPM!`;
      input.disabled = true;
    }
  });
</script>
```

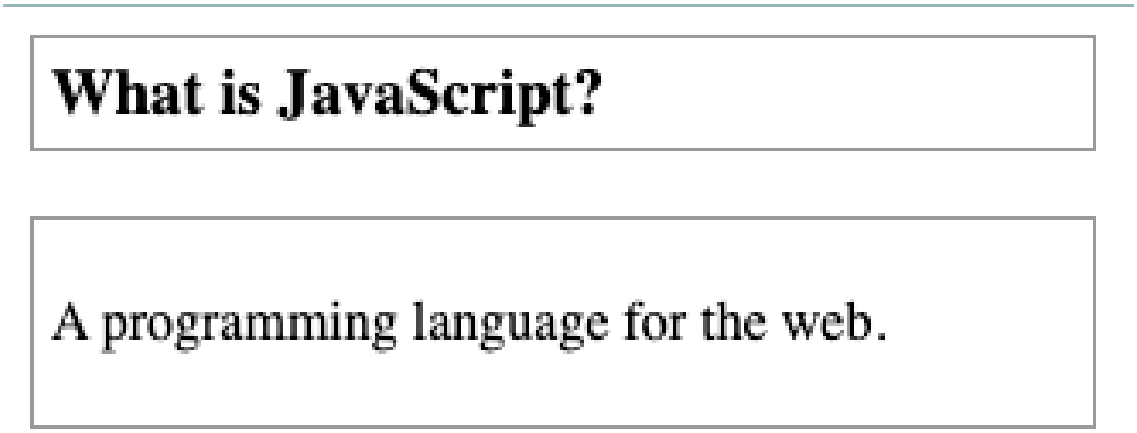


12. Accordion Menu

Description: Collapsible sections that reveal or hide content when clicked—great for FAQs.

Tools & Concepts: Class toggling, CSS transitions, event listeners

```
<style>
  .content {
    display: none;
  }
  .active .content {
    display: block;
  }

  h3,
  .content {
    width: 300px;
    border: solid 1px #999999;
    padding: 5px;
  }
</style>

<div class="accordion" onclick="this.classList.toggle('active')">
  <h3>What is JavaScript?</h3>
  <div class="content"><p>A programming language for the web.</p></div>
</div>
```

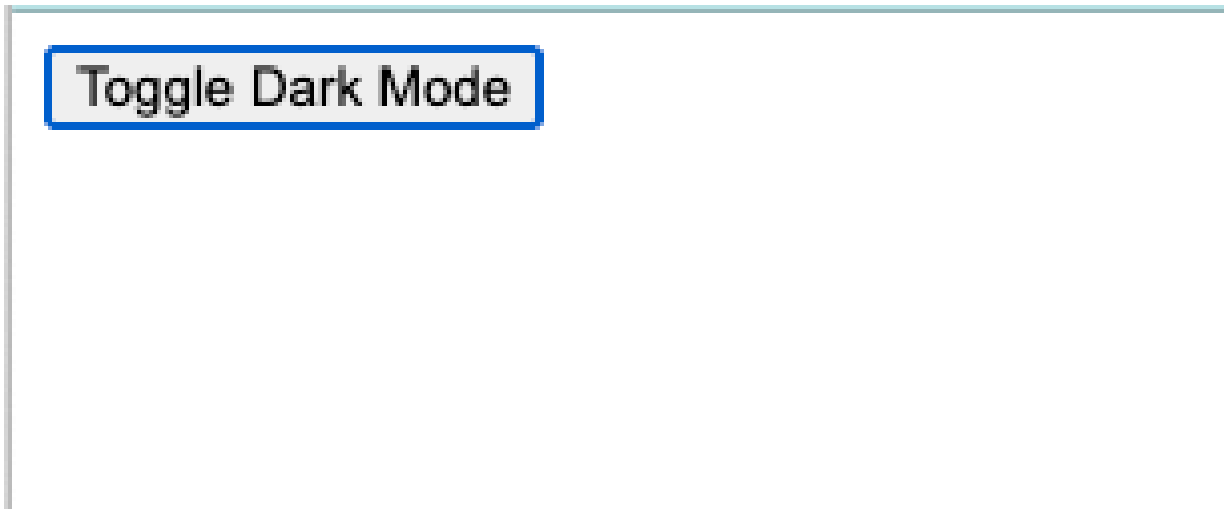


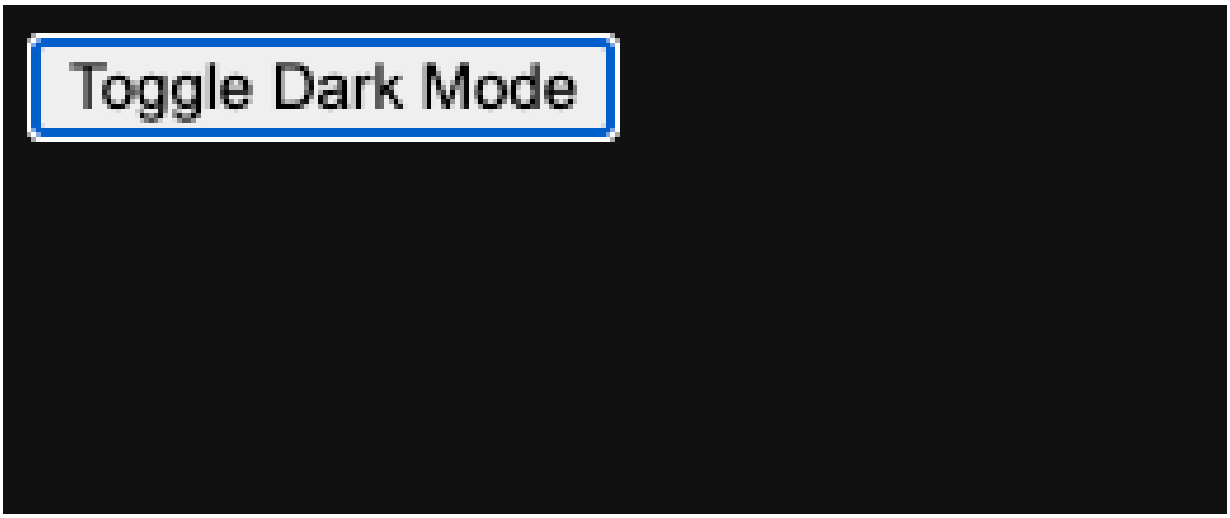
13. Dark/Light Mode Toggle

Description: Adds a toggle button to switch between dark and light themes dynamically.

Tools & Concepts: DOM class manipulation, `localStorage`

```
<button onclick="toggleMode()">Toggle Dark Mode</button>

<script>
  function toggleMode() {
    document.body.classList.toggle("dark");
    localStorage.setItem("theme", document.body.classList.contains("dark") ?
"dark" : "light");
  }

  if (localStorage.getItem("theme") === "dark") {
    document.body.classList.add("dark");
  }
</script>

<style>
  body.dark {
    background: #111;
    color: #fff;
  }
</style>
```



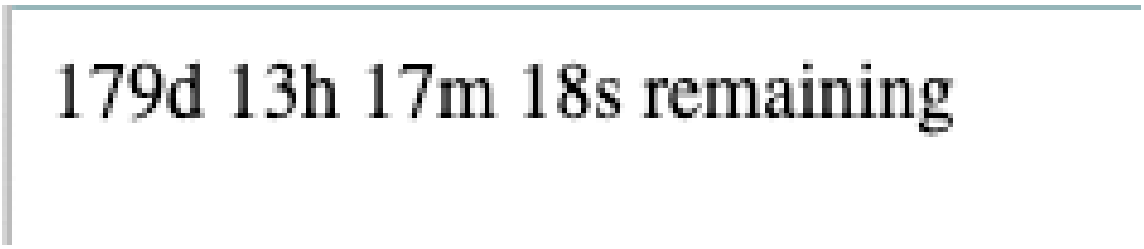


14. Countdown Timer

Description: A timer that counts down to a specific future date/time.

Tools & Concepts: `setInterval`, `Date`, time difference calculation

```
<p id="countdown"></p>

<script>
  const target = new Date("2025-12-31T23:59:59");

  setInterval(() => {
    const now = new Date();
    const diff = target - now;

    if (diff <= 0) {
      document.getElementById("countdown").textContent = "🎉 Happy New Year!";
      return;
    }

    const days = Math.floor(diff / (1000 * 60 * 60 * 24));
    const hours = Math.floor((diff / (1000 * 60 * 60)) % 24);
    const minutes = Math.floor((diff / (1000 * 60)) % 60);
    const seconds = Math.floor((diff / 1000) % 60);

    document.getElementById("countdown").textContent =
      `${days}d ${hours}h ${minutes}m ${seconds}s remaining`;
  }, 1000);
</script>
```

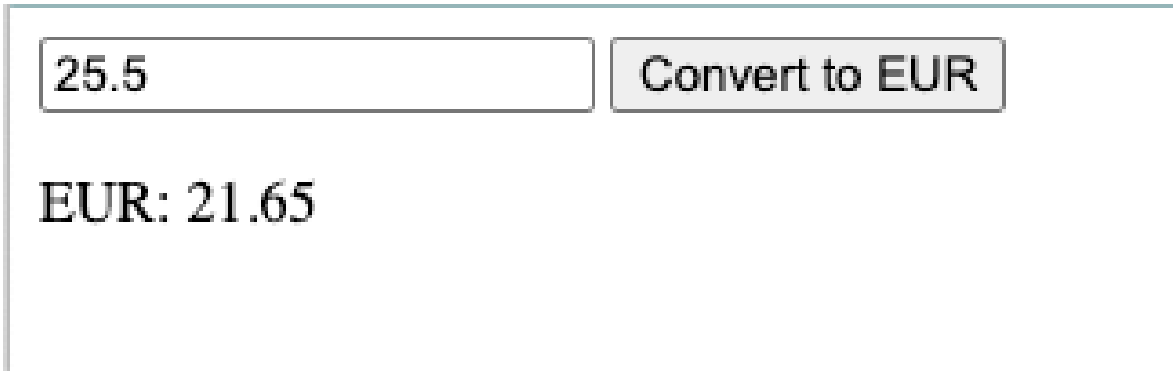


15. Currency Converter

Description: Converts amounts between two currencies using exchange rates from an API like exchangerate-api.com.

Tools & Concepts: Fetch API, JSON, math operations

```
<input id="usd" placeholder="USD amount" />
<button onclick="convert()">Convert to EUR</button>
<p id="eur"></p>

<script>
  async function convert() {
    const amount = document.getElementById("usd").value;
    const res = await fetch("https://api.exchangerate-api.com/v4/latest/USD");
    const data = await res.json();
    const rate = data.rates.EUR;
    document.getElementById("eur").textContent = `EUR: ${(amount *
rate).toFixed(2)}`;
  }
</script>
```

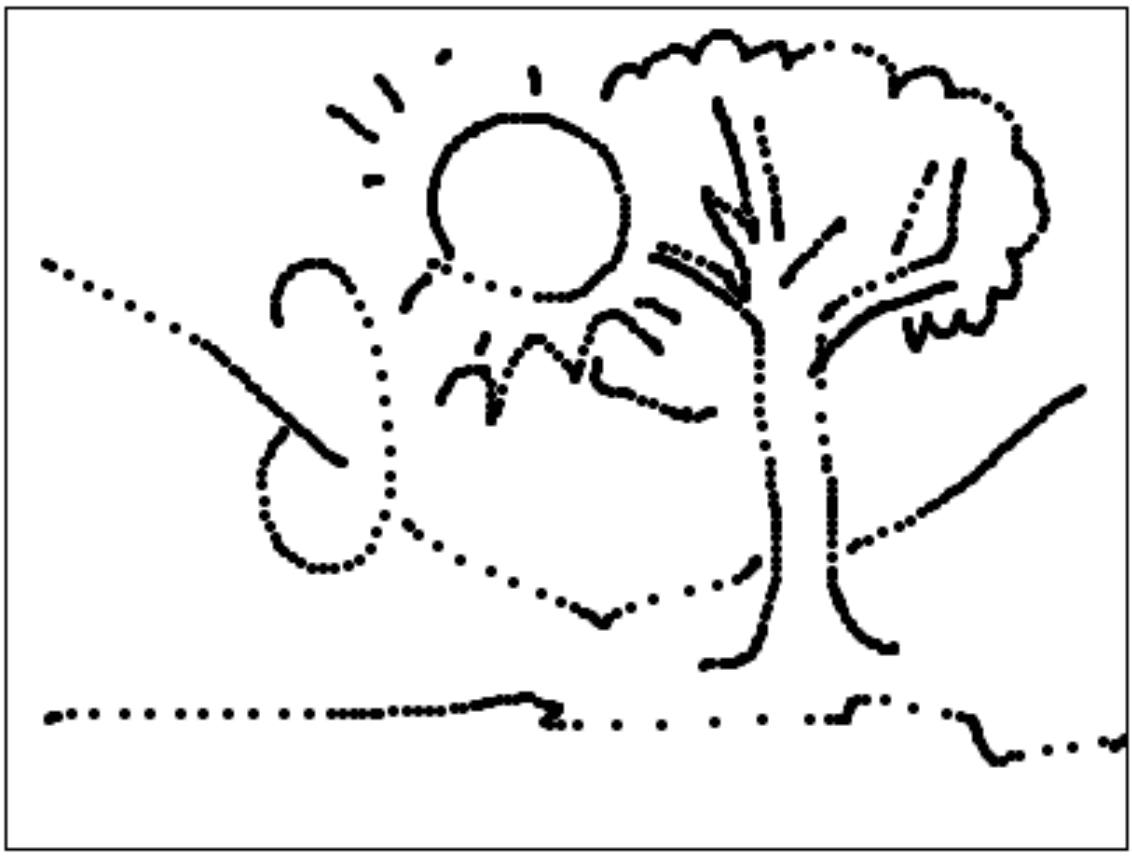


16. Simple Drawing App (Paint Clone)

Description: A mini canvas app where users can draw freehand using the mouse.

Tools & Concepts: Canvas API, mouse events

```
<canvas id="draw" width="400" height="300" style="border:1px solid #000;">
</canvas>

<script>
  const canvas = document.getElementById("draw");
  const ctx = canvas.getContext("2d");
  let drawing = false;

  canvas.addEventListener("mousedown", () => drawing = true);
  canvas.addEventListener("mouseup", () => drawing = false);
  canvas.addEventListener("mousemove", draw);

  function draw(e) {
    if (!drawing) return;
    ctx.fillStyle = "#000";
    ctx.beginPath();
    ctx.arc(e.offsetX, e.offsetY, 2, 0, Math.PI * 2);
    ctx.fill();
  }
</script>
```

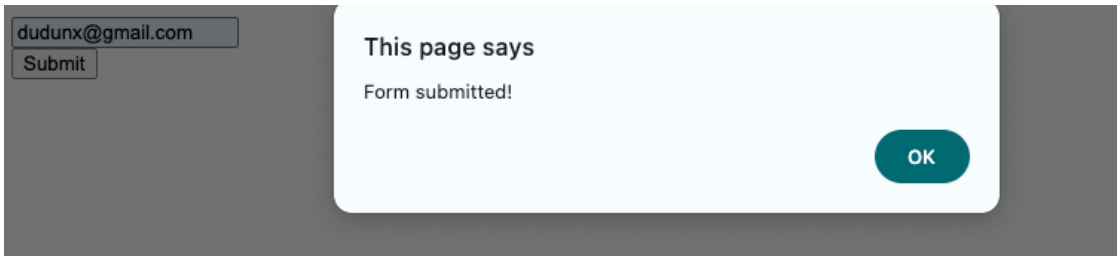


17. Form Validation

Description: Validates a basic registration form with JavaScript before submission.

Tools & Concepts: Form events, regex, validation feedback

```
<form onsubmit="return validateForm()">
  <input id="email" placeholder="Enter your email" />
  <span id="error" style="color:red;"></span><br/>
  <button type="submit">Submit</button>
</form>

<script>
  function validateForm() {
    const email = document.getElementById("email").value;
    const error = document.getElementById("error");
    const regex = /^[^\s@]+@[^\s@]+\.[^\s@]+$/;

    if (!regex.test(email)) {
      error.textContent = "Invalid email address.";
      return false;
    }

    error.textContent = "";
    alert("Form submitted!");
    return true;
  }
</script>
```

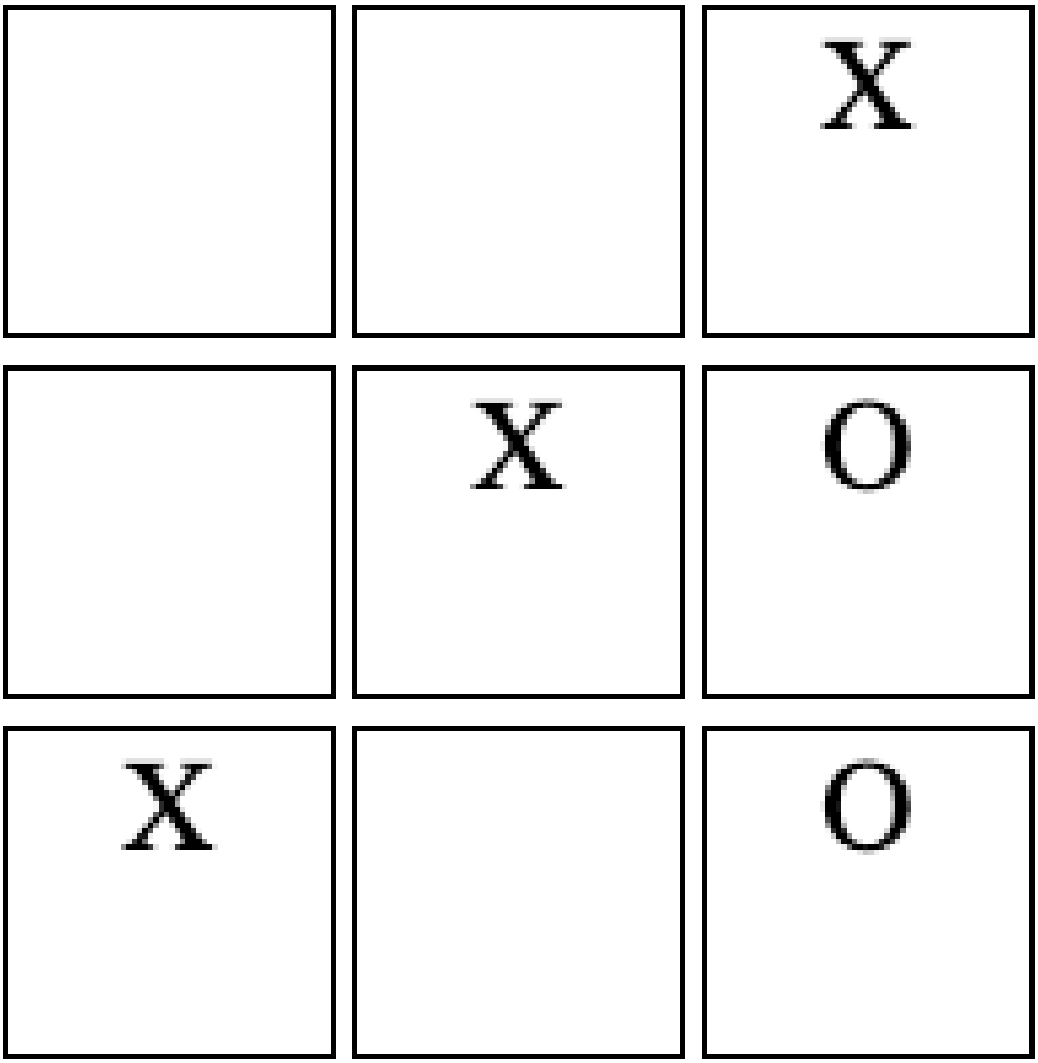


18. Tic-Tac-Toe Game

Description: A simple two-player Tic-Tac-Toe board with win checking.

Tools & Concepts: Grid layout, click events, game logic

```
<style>
  .grid { display: grid; grid-template-columns: repeat(3, 60px); gap: 5px; }
  .cell { width: 60px; height: 60px; text-align: center; font-size: 24px; cursor:
pointer; border: 1px solid #000; }
</style>

<div class="grid" id="board"></div>
<p id="status"></p>

<script>
  const board = document.getElementById("board");
  const status = document.getElementById("status");
  let current = "X";
  const cells = Array(9).fill("");

  for (let i = 0; i < 9; i++) {
    const cell = document.createElement("div");
    cell.className = "cell";
    cell.onclick = () => mark(i, cell);
    board.appendChild(cell);
  }

  function mark(i, cell) {
    if (cells[i]) return;
    cells[i] = current;
    cell.textContent = current;
    if (checkWin()) {
      status.textContent = `${current} wins!`;
      board.style.pointerEvents = "none";
    } else {
      current = current === "X" ? "O" : "X";
    }
  }

  function checkWin() {
    const winPatterns = [
      [0,1,2],[3,4,5],[6,7,8],
      [0,3,6],[1,4,7],[2,5,8],
      [0,4,8],[2,4,6]
    ];
    return winPatterns.some(p => p.every(i => cells[i] === current));
  }
</script>
```

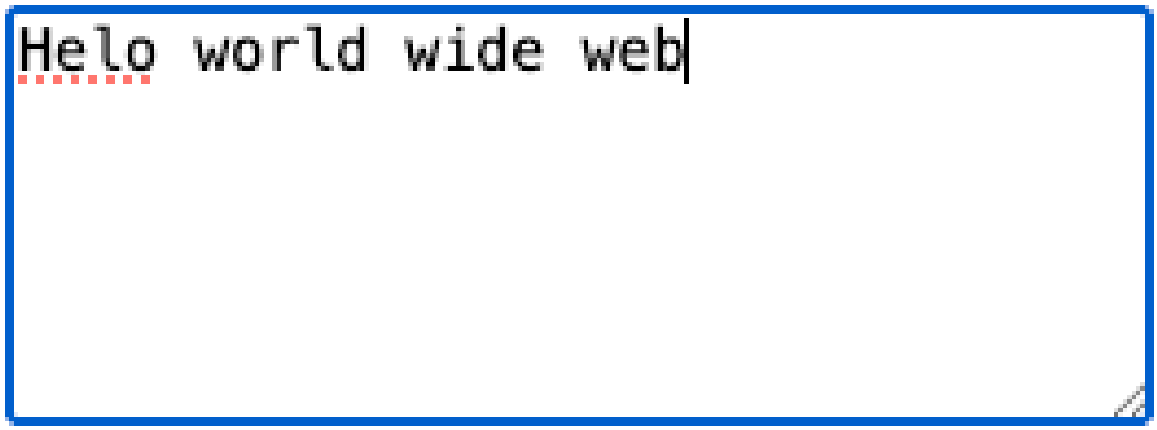


19. LocalStorage Note App

Description: Let users type notes that persist even after refreshing the browser.

Tools & Concepts: localStorage, textarea, events

```
<textarea id="note" rows="6" cols="30" placeholder="Type your note here...">
</textarea>

<script>
  const note = document.getElementById("note");
  note.value = localStorage.getItem("note") || "";

  note.addEventListener("input", () => {
    localStorage.setItem("note", note.value);
  });
</script>
```



20. Scroll to Top Button

Description: A floating button that appears after scrolling and takes the user back to the top of the page.

Tools & Concepts: Scroll events, smooth behavior

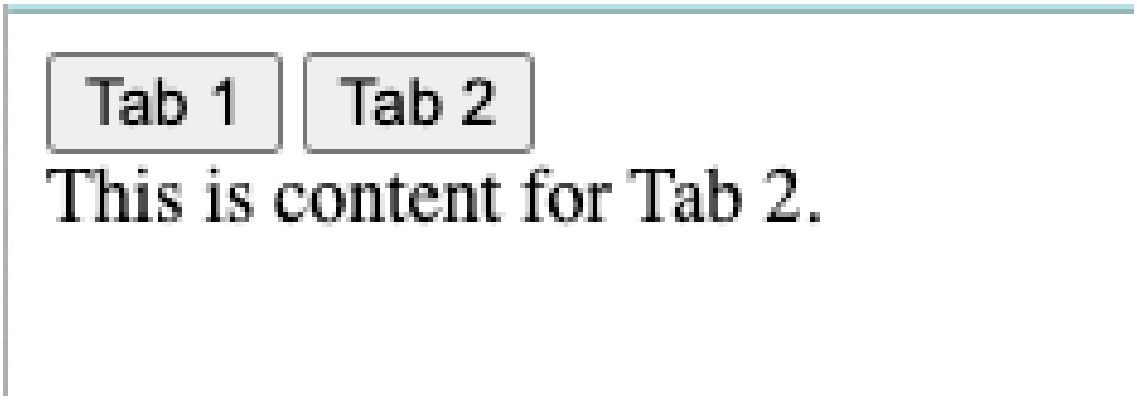
```
<button onclick="topFunction()" id="topBtn"
style="display:none;position:fixed;bottom:20px;right:20px;">⬆️ Top</button>

<script>
  const topBtn = document.getElementById("topBtn");

  window.onscroll = () => {
    topBtn.style.display = window.scrollY > 100 ? "block" : "none";
  };

  function topFunction() {
    window.scrollTo({ top: 0, behavior: "smooth" });
  }
</script>
```

21. Tabs Component

Description: Creates a tabbed interface where clicking a tab shows its content while hiding others.

Tools & Concepts: Event delegation, class manipulation

```
<div>
  <button onclick="showTab(0)">Tab 1</button>
  <button onclick="showTab(1)">Tab 2</button>
</div>

<div class="tab-content">This is content for Tab 1.</div>
<div class="tab-content" style="display:none;">This is content for Tab 2.</div>

<script>
  function showTab(index) {
    const tabs = document.querySelectorAll(".tab-content");
    tabs.forEach((tab, i) => tab.style.display = i === index ? "block" : "none");
  }
</script>
```



22. Stopwatch

Description: A simple stopwatch with Start, Stop, and Reset buttons.

Tools & Concepts: setInterval, state management

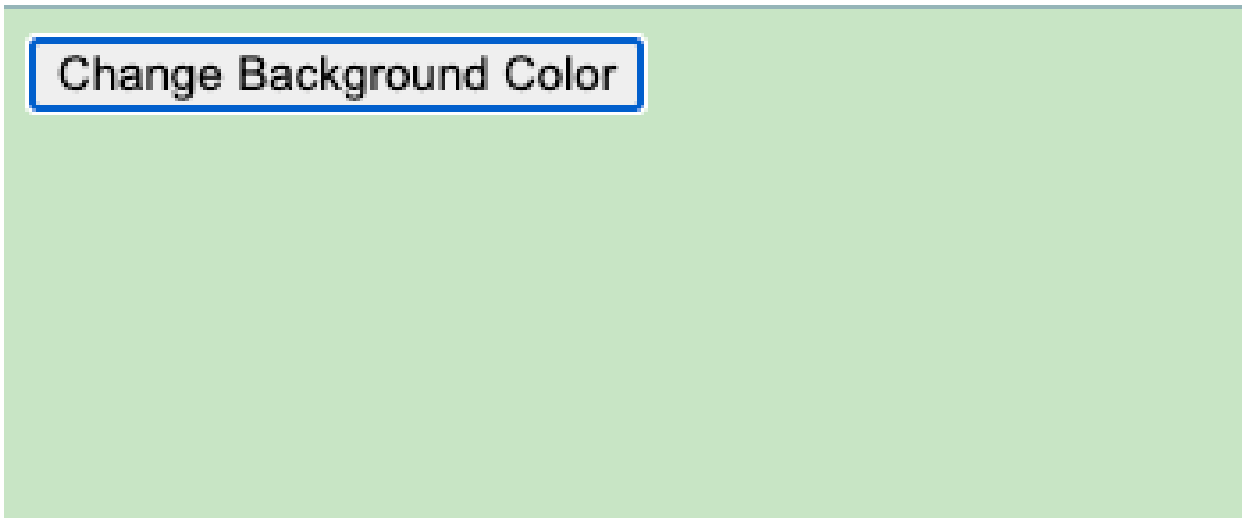
```
<div class="stopwatch"><p id="display">0.00</p></div>
<button onclick="start()">Start</button>
<button onclick="stop()">Stop</button>
<button onclick="reset()">Reset</button>

<style>
  p {
    font-size: 32px;
    font-family: "Courier New", Courier, monospace;
    font-weight: bold;
    color: yellow;
  }
  button {
    margin-left: 10px;
  }
  .stopwatch {
    margin: 10px;
    padding: 10px;
    background-color: black;
    width: 150px;
  }
</style>

<script>
  let time = 0,
    interval;
  function start() {
    if (interval) return;
    interval = setInterval(() => {
      time += 0.01;
      document.getElementById("display").textContent = time.toFixed(2);
    }, 10);
  }
  function stop() {
    clearInterval(interval);
    interval = null;
  }
  function reset() {
    stop();
    time = 0;
    document.getElementById("display").textContent = "0.00";
  }
</script>
```

23. Random Color Generator

Description: Generates and applies a random background color to the body when clicked.

Tools & Concepts: Math.random, DOM style manipulation

```
<button onclick="generateColor()">Change Background Color</button>

<script>
  function generateColor() {
    const color = "#" + Math.floor(Math.random() * 16777215).toString(16);
    document.body.style.backgroundColor = color;
  }
</script>
```

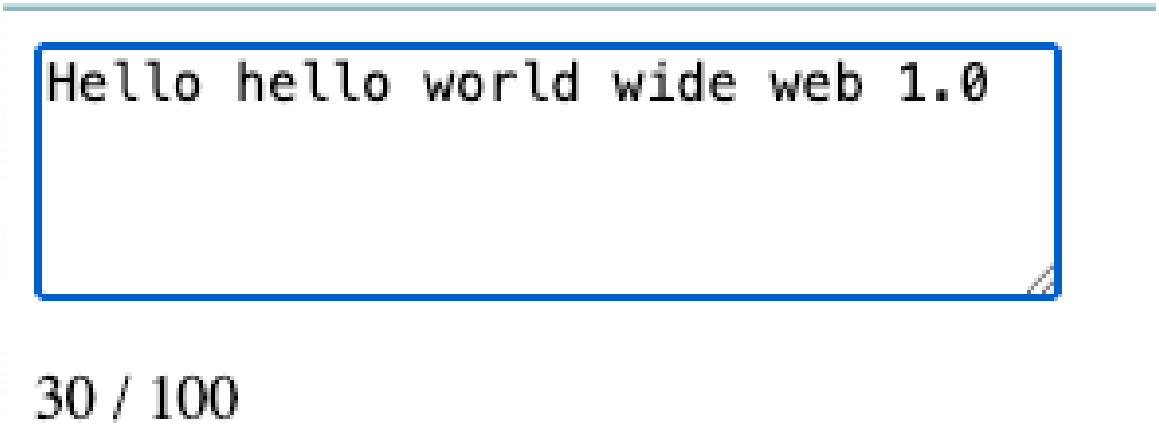


24. Character Counter

Description: Displays live character count of an input field and limits it to a max length.

Tools & Concepts: input events, length tracking

```
<textarea id="text" maxlength="100" rows="4" cols="30"></textarea>
<p id="count">0 / 100</p>

<script>
  const text = document.getElementById("text");
  const count = document.getElementById("count");

  text.addEventListener("input", () => {
    count.textContent = `${text.value.length} / 100`;
  });
</script>
```



25. Progress Bar on Scroll

Description: Shows a progress bar at the top indicating how much of the page has been scrolled.